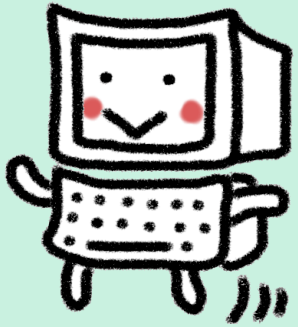


Project 2

OpenFlow Protocol Observation &
Flow Rule Installation

Deadline: 2021/10/13 (WED) 23:59

TA: 陳志祥

Outline

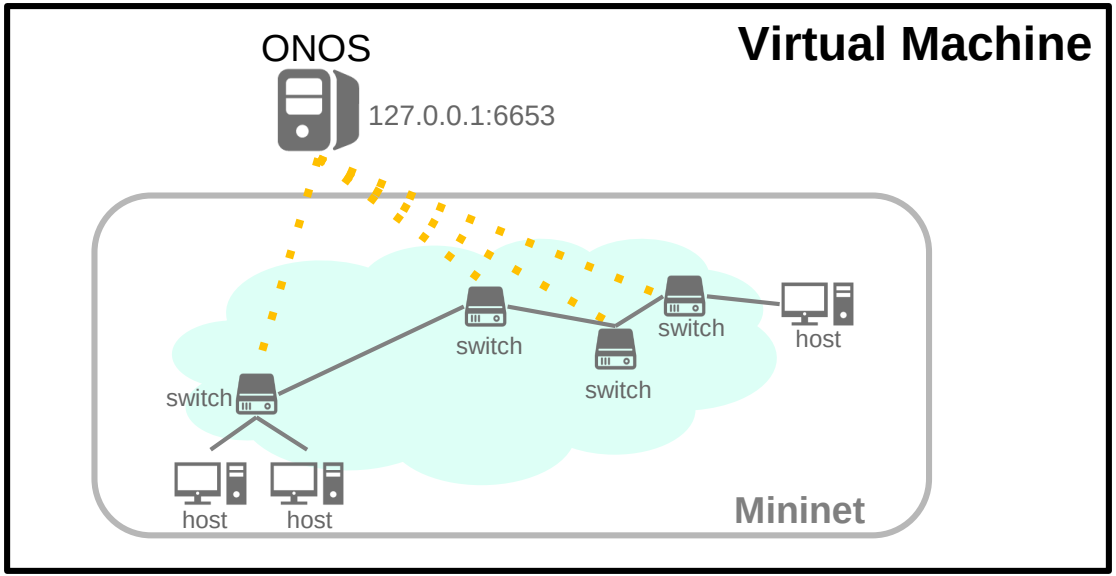
- OpenFlow Messages
  - Monitor traffic between ONOS & Switches
  - OpenFlow Message Observation
- Install/ Delete Flow Rules
  - Curl
  - ONOS and Topology Setup
  - Method 1: via Command “curl”
  - Method 2: via ONOS Web GUI
- Project 2 Requirements
  - Part 1: Answer Questions
  - Part 2: Install Flow Rules
  - Part 3: Create Broadcast Storm
  - Part 4: Trace ReactiveForwarding

Outline

- OpenFlow Messages
  - Monitor traffic between ONOS & Switches
  - OpenFlow Message Observation
- Install/ Delete Flow Rules
- Project 2 Requirements

OpenFlow Protocol

- ONOS SDN controller uses OpenFlow messages to communicate with OVS switches.
  - Packet-in/out message, Flow install/remove message, hello, etc.



Wireshark Installation

- Wireshark is an open-source and widely-used network packet analyzer
  - Can capture packets on any specified interface
- Installation steps:
  1. Download package information

```
$ sudo apt update                    # update all packages information
```

  1. Install Wireshark

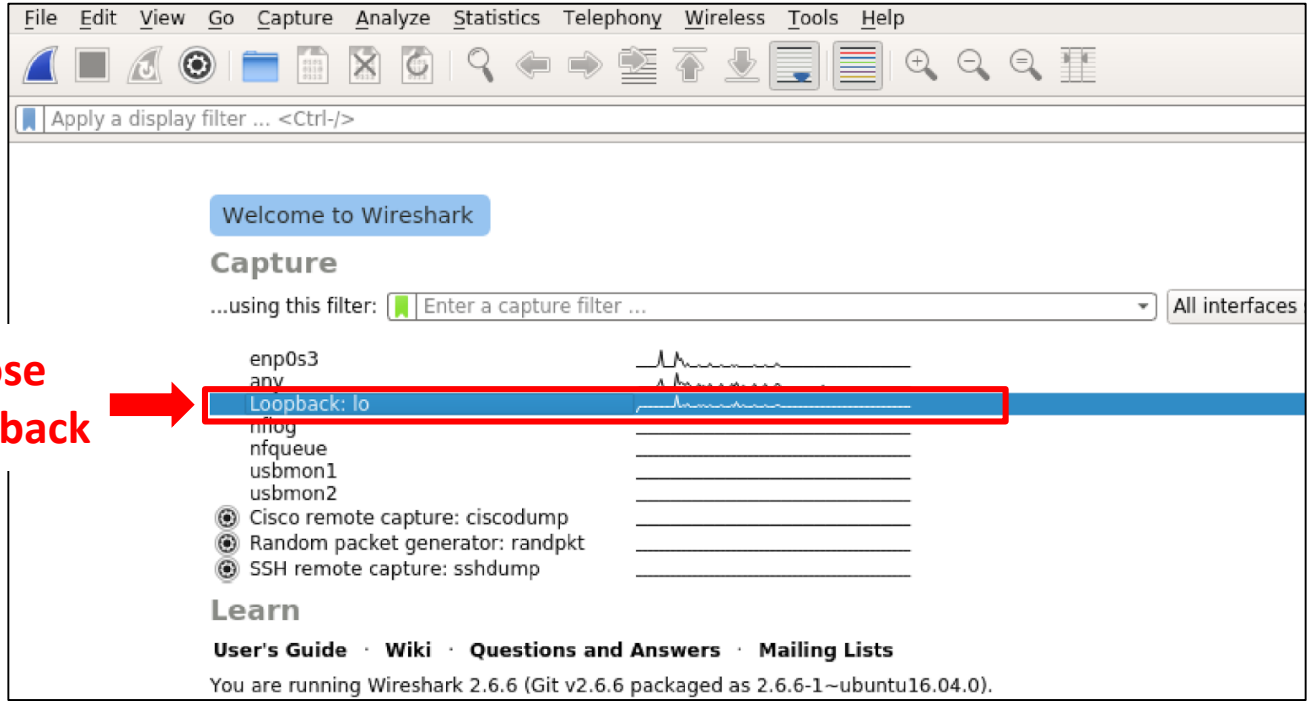
```
$ sudo apt install wireshark
```

- Start Wireshark

```
$ sudo wireshark
```

Capture Packets in Wireshark

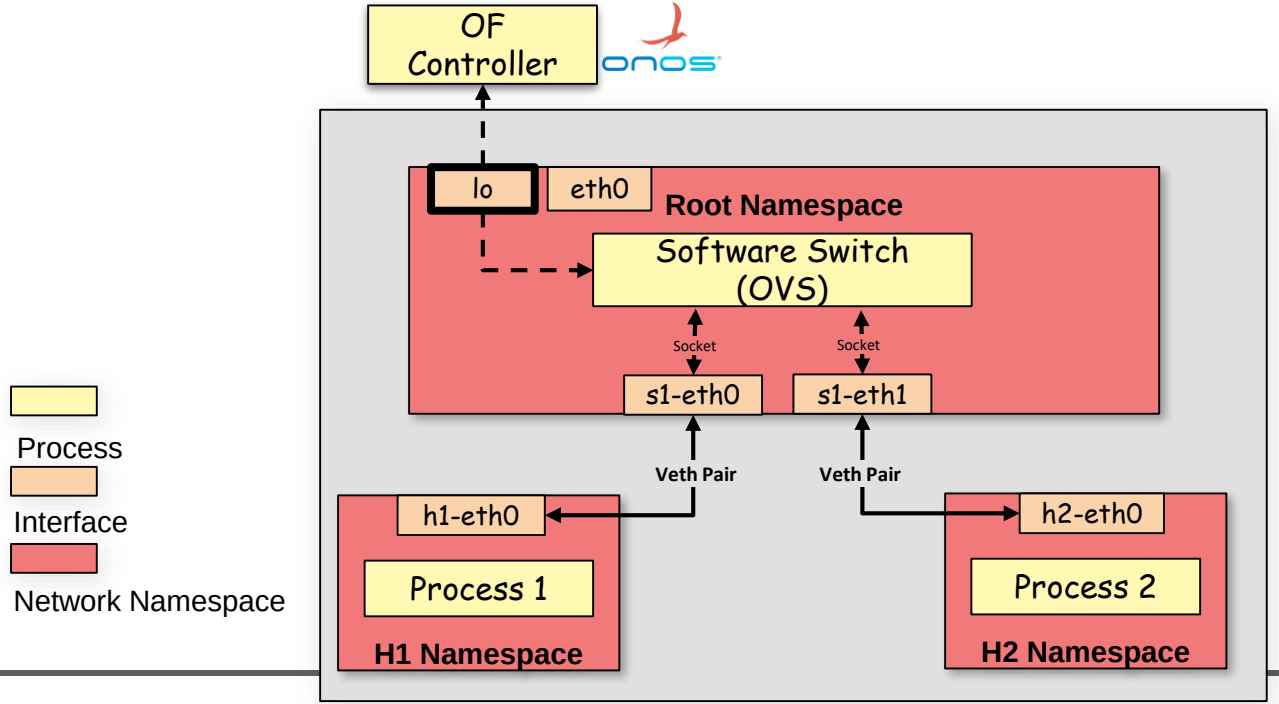
- Both ONOS and Mininet locally run in VM
  - We can capture packets on the Loopback (lo) interface



Choose Loopback

Mininet and Network Namespace

- Mininet utilizes **network namespace** to emulate networks
  - OVS runs in root network namespace
  - Each host runs in its own network namespace



Outline

- **OpenFlow Messages**
  - Monitor traffic between ONOS & Switches
  - **OpenFlow Message Observation**
- Install/ Delete Flow Rules
- Project 2 Requirements

Capturing OpenFlow Messages

1. Start ONOS
2. Activate ReactiveForwarding

```
onos> apps –a –s                    # (optional) check activated application
onos> app activate fwd              # activate ReactiveForwarding
```

3. Start Mininet with default topology

```
$ sudo mn --controller=remote,127.0.0.1:6653
```

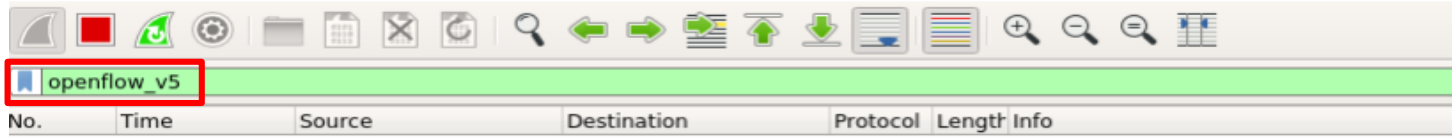
4. Ping a host in Mininet

```
mininet> h1 ping h2 -c 5            # send five ICMP echo_reqest packets
```

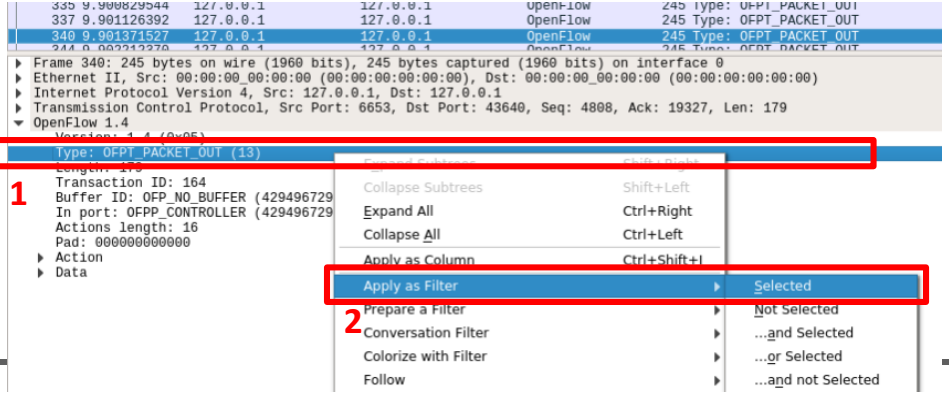
5. Exit Mininet and stop capturing packets in Wireshark when ping terminates
6. Observe captured OpenFlow packets

Filter in Wireshark

- Use keyword “openflow_v5” to filter **OpenFlow v1.4.0** packets
  - ONOS v2.2.0 uses Openflow v1.4.0



  - Alternatively, apply filter in the following steps:
    1. Right click on the packet header field which you want to apply as filter
    2. Choose “Apply as Filter” and click “Selected”
    3. Wireshark will immediately filter out all the relevant packets



Outline

- OpenFlow Messages
- **Install/ Delete Flow Rules**
  - **Curl**
  - **ONOS and Topology Setup**
  - **Method 1: via Command “curl”**
  - **Method 2: via ONOS Web GUI**
- Project 2 Requirements

Curl–Command Tool For Transferring Data

- Command format

```
curl [options] [URL...]
```

- Transferring data with URL

```
$ curl -u <user:password> -X <method-type> -H <header> -d <data> [URL...]
                  # option "-X" specifies a HTTP request method
                  # option “-H” includes extra header in the HTTP request
                  # option “-d” sends specified data in a POST request
                  # URL (Uniform Resource Locator)
```

  - "<data>" can be a file name with prefix “@”

```
$ curl -u <user:password> -X <method-type> -H <header> -d @<filename> [URL...]
```

- Manpage for command "curl"
  - http://manpages.ubuntu.com/manpages/xenial/man1/curl.1.html

Outline

- OpenFlow Messages
- **Install/ Delete Flow Rules**
  - Curl
  - **ONOS and Topology Setup**
  - Method 1: via Command “curl”
  - Method 2: via ONOS Web GUI
- Project 2 Requirements

ONOS & Topology Setup

1. Restart ONOS
   a) <ctrl+c> in the ONOS log panel to shutdown the ONOS instance
   b) Start ONOS

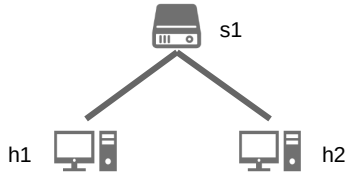
```
demo@SDN-NFV:~/onos$ ok clean
    # ok is an alias of command “bazel run onos-local -- ”
```

2. Deactivate ReactiveForwarding APP

```
onos> app deactivate fwd    # deactivate ReactiveForwarding
```

3. Start Mininet with default (minimal) topology

```
$ sudo mn --controller=remote,127.0.0.1:6653
```



4. Make sure that two hosts **CAN NOT** ping each other

```
mininet> h1 ping h2
```

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
```

Outline

- OpenFlow Messages
- **Install/ Delete Flow Rules**
  - REST & curl
  - ONOS and Topology Setup
  - **Method 1: via Command “curl”**
  - Method 2: via ONOS Web GUI
- Project 2 Requirements

Create a JSON file of flow rules

- Example: JSON file for a flow rule

flows1.json

```
{
   "priority": 50000,
   "timeout": 0,
   "isPermanent": true,
   "selector": {
      "criteria": [
         {
            "type": "IN_PORT",
            "port": 1
         }
      ]
   },
   "treatment": {
      "instructions": [
         {
            "type": "OUTPUT",
            "port": 2
         }
      ]
   }
}
```

JSON File: Match Fields

```
{
    "priority": 50000,
    "timeout": 0,
    "isPermanent": true,
    "selector": {
        "criteria": [
            {
                "type": "IN_PORT",
                "port": 1
            }
        ]
    },
    "treatment": {
        "instructions": [
            {
                "type": "OUTPUT",
                "port": 2
            }
        ]
    }
}
```

flows1.json

```
"selector": {
    "criteria": [
        {
            "type": "IN_PORT",
            "port": 1
        },
        { … },
        …
    ]
},
```

JSON File: Actions

```
{
    "priority": 50000,
    "timeout": 0,
    "isPermanent": true,
    "selector": {
        "criteria": [
            {
                "type": "IN_PORT",
                "port": 1
            }
        ]
    },
    "treatment": {
        "instructions": [
            {
                "type": "OUTPUT",
                "port": 2
            }
        ]
    }
}
```

flows1.json

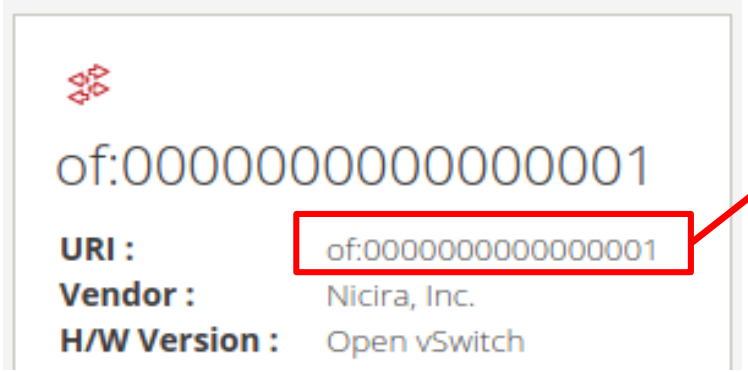
```
"treatment": {
    "instructions": [
        {
            "type": "OUTPUT",
            "port": 2
        },
        { … },
        …
    ]
}
```

Upload JSON File to ONOS

- Install flow rules on ONOS with JSON file (**flows1.json**)

```
$ curl -u onos:rocks \
> -X POST \
> -H 'Content-Type: application/json' \
> -d @flows1.json \
> 'http://localhost:8181/onos/v1/flows/of:0000000000000001'
```

Device ID



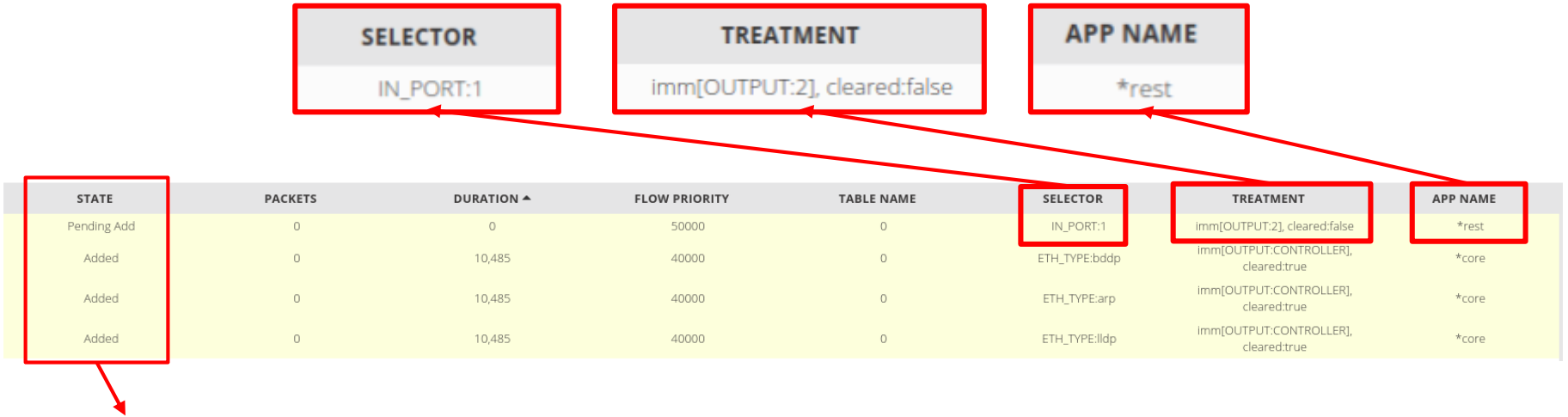
- deviceId **MUST be** the URI shown in the ONOS web GUI
- deviceId is set by either ONOS or user specified topology file (i.e. *.py)

Check whether the flow rule is installed

1. Go to ONOS web GUI (http://localhost:8181/onos/ui)
2. Left click on  . Then, the panel of switch info will pop out
3. Left click on 

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	36	50000	0	IN_PORT:1	imm[OUTPUT:2], cleared:false	*rest
Added	0	960	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	960	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	12	960	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core

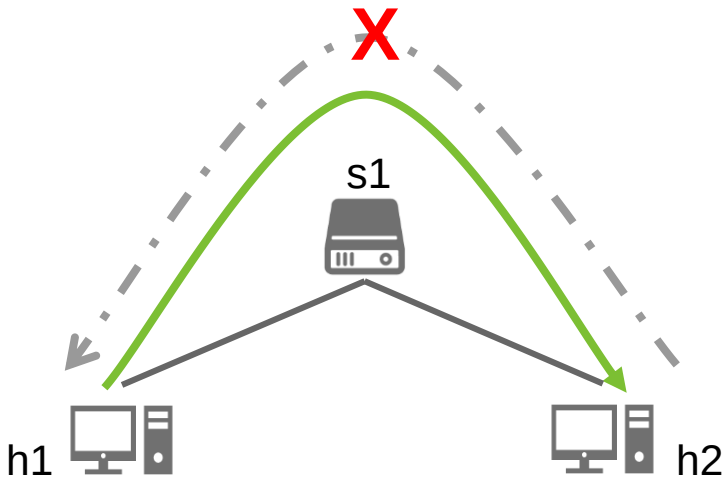
Check whether the flow rule is installed (Cont.)



- Flow Rule state:
  - **PENDING_ADD**—this indicates that ONOS has received a request from the application to install the flow rule, but that flow has NOT yet been observed on the device.
  - **ADDED**—once the flow rule subsystem observes the flow on the device it will transition to this state.

Why Hosts Still Can't Ping Each Other?

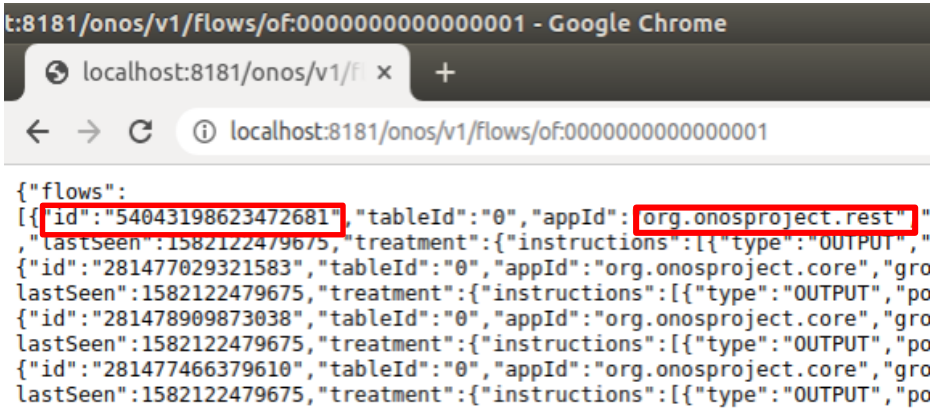
- Because we have **only** installed a flow rule for one direction
  - s1 can forward packets from h1 to h2
  - But, s1 CANNOT forward packets from h2 to h1
    - By default, s1 drops a packet if the packet does not match any flow rule (i.e. **table-miss**)



Delete Flow Rules

- Use URL to find the **ID** of particular flow rules

  Ex. http://localhost:8181/onos/v1/flows/of:0000000000000001



  - ID of the flow we just added is **54043198623472681**

- Alternatively,, we could use “curl” to get flow information

```
$ curl -u onos:rocks -X GET -H ‘Accept: application/json’ \
> ‘http://localhost:8181/onos/v1/flows/of:0000000000000001’
```

Delete Flow Rules (Cont.)

- Then, delete the flow rule with flowID 5404319862347268l

```
$ curl -u onos:rocks \
> -X DELETE \
> -H 'Accept: application/json' \
> 'http://localhost:8181/onos/v1/flows/of:0000000000000001 \
> /5404319862347268l'
```

Device ID

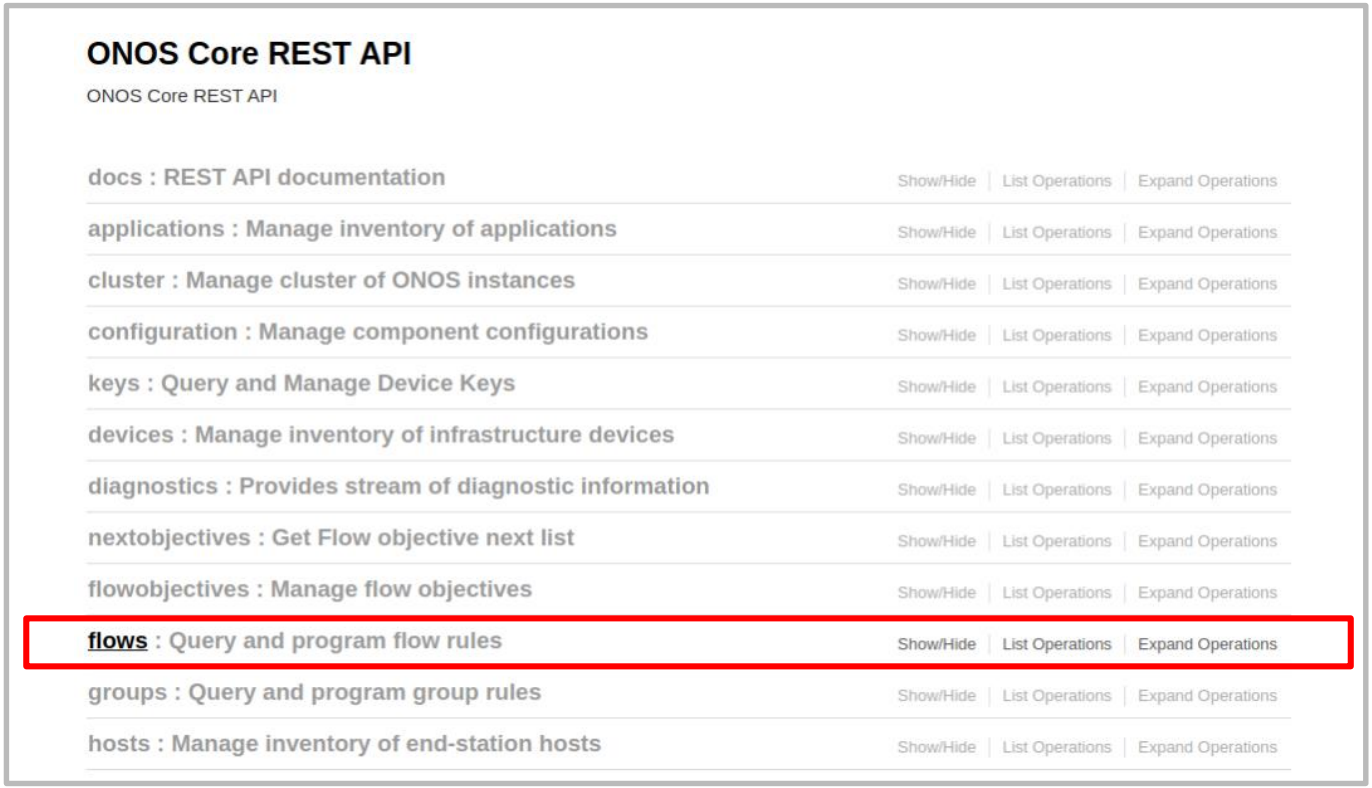
Flow ID

Outline

- OpenFlow Messages
- **Install/ Delete Flow Rules**
  - curl
  - ONOS and Topology Setup
  - Method 1: Via Command “curl”
  - **Method 2: Via ONOS Web GUI**
- Project 2 Requirements

REST API on ONOS Web GUI

- Browse http://localhost:8181/onos/v1/docs



Using Web GUI to Install Flow Rule

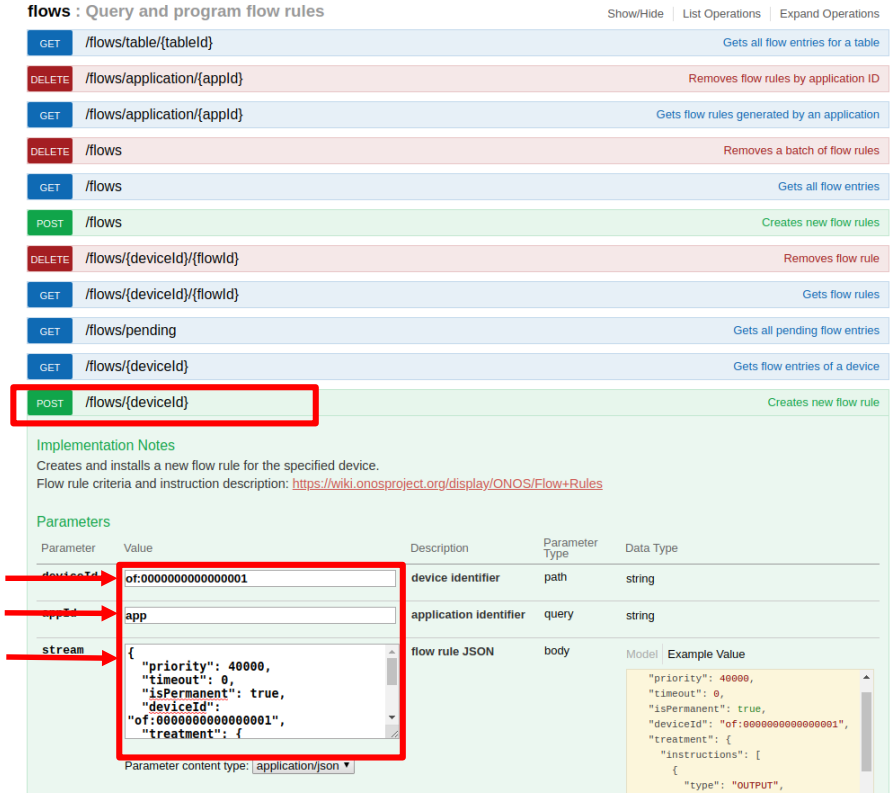
- Fill out required fields ("appId" could be arbitrary string)

Select

POST /flows/{deviceId}

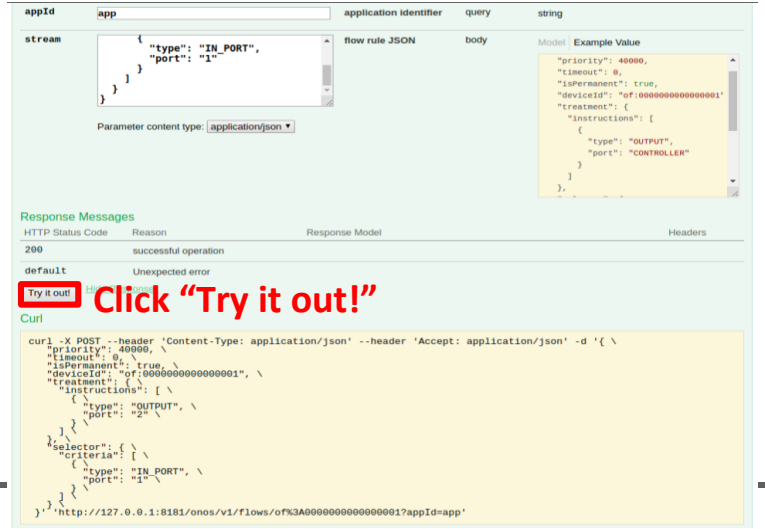
Type In

deviceId
appId
JSON file



Transfer Flow Rule on Web GUI

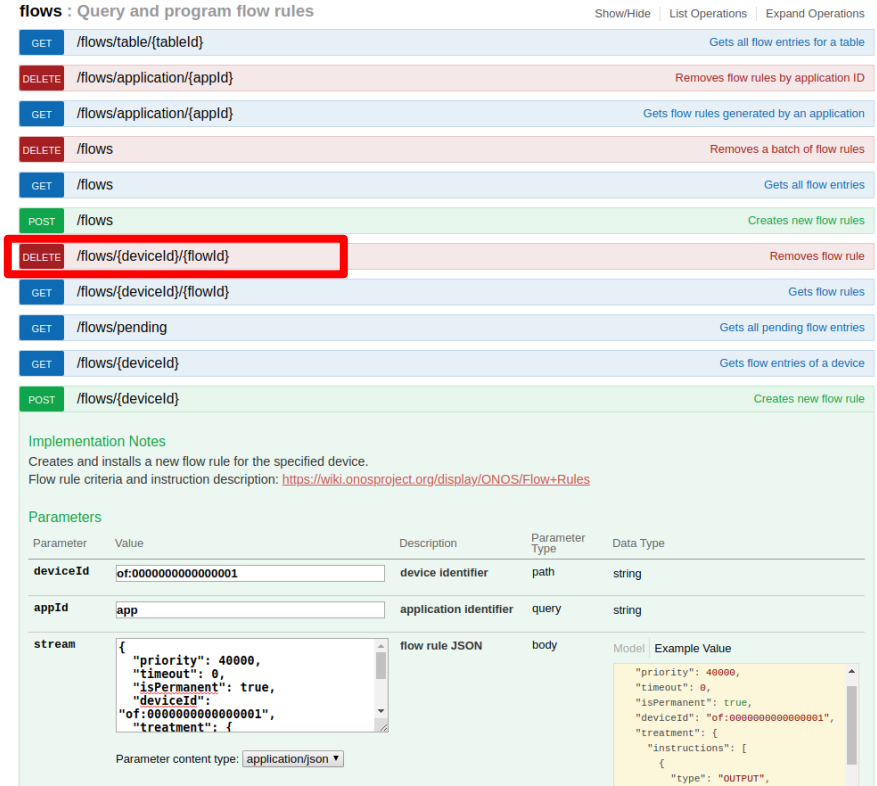
- Click “Try it out!”
  - Web will pass the JSON stream to ONOS
  - Status code 201 represents HTTP Request is granted
  - In case of “curl”, use “-i” option to include HTTP Response header in the output



HTTP response replied by ONOS

Delete Flow Rule via ONOS Web GUI

- Same procedure as installing flow rules



Outline

- OpenFlow Messages
- Install/ Delete Flow Rules
- **Project 2 Requirements**
  - **Part 1: Answer Questions (30%)**
  - **Part 2: Install Flow Rules (30%)**
  - **Part 3: Create Broadcast Storm (30%)**
  - **Part 4: Trace ReactiveForwarding (10%)**

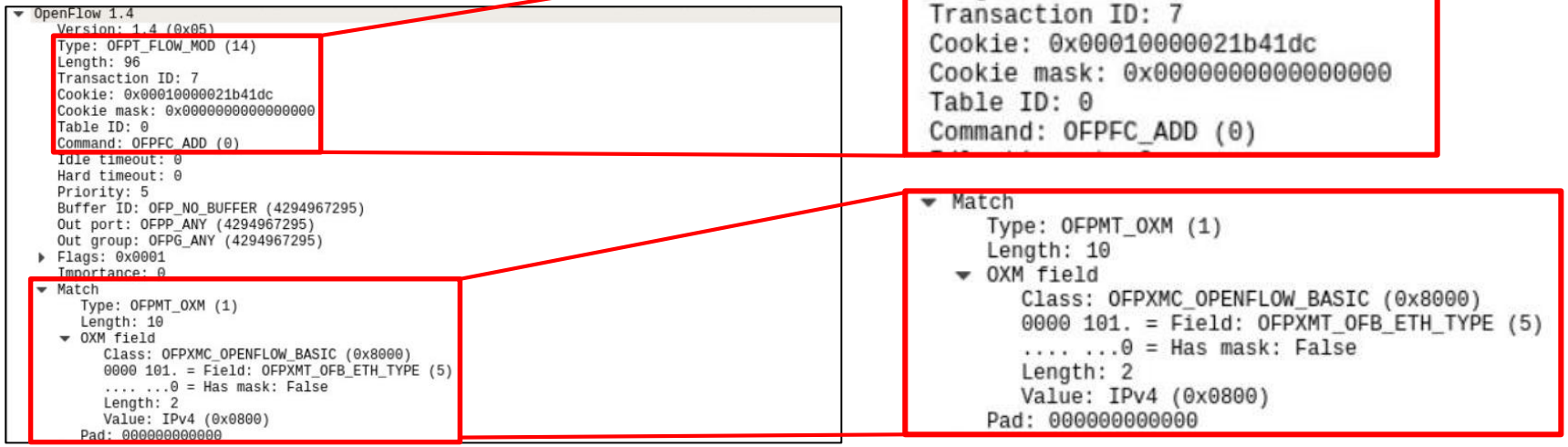
Part 1: Answer Questions (1/2)

- Preparation:
  1. Start capturing packets on the loopback interface (lo) with Wireshark.
  2. Create a topology mentioned before (i.e. default topology).
  3. Activate “org.onosproject.fwd”.
  4. Execute command “h1 ping h2 -c 5” in Mininet CLI.
  5. Exit Mininet and stop capturing packets once ping terminates.

- Questions:
  1. How many **OpenFlow headers** with type “OFPT_FLOW_MOD” and command “OFPFC_ADD” are there among all the packets?
  2. What are the **match fields** and the corresponding **actions** in each “OFPT_FLOW_MOD” message?
  3. What are the value of **timeout** for each flow rule installed in s1?

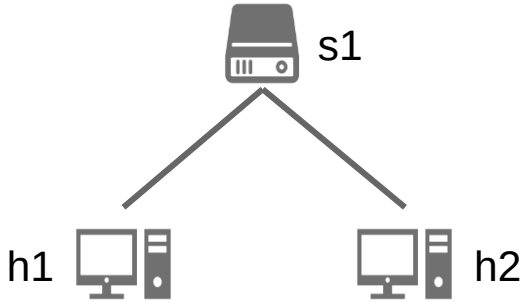
Part 1: Answer Questions (2/2)

- Hints
  - A single packet may contain multiple OpenFlow headers
  - Only count the number of **distinct** OpenFlow headers
    - If match fields of two headers are the same, just count once
  - Value of timeout can be zero



Part 2: Install Flow Rules (1/3)

- Please deactivate all the apps, **except those** initially activated.
  “org.onosproject.hostprovider”,
  “org.onosproject.lldpprovider”,
  “org.onosproject.optical-model”,
  “org.onosproject.openflow-base”,
  “org.onosproject.openflow”,
  “org.onosproject.drivers”
  and “org.onosproject.gui2”.
- Use the following topology (i.e. h1-s1-h2):



- Hand in all your flow rule files (.json)

Note: Host1 should be able to ping host2 if you install the flow rules correctly.

Part 2: Install Flow Rules (2/3)

- Install following flow rules to forward ARP packets
  - Match Fields
    - Ethernet type (ARP)
  - Actions
    - Output from port, forwarding ARP packets to hosts
- Verify the flow rules your installed

```
mininet> h1 arping h2                # send ARP request
```

```
mininet> h1 arping h2
ARPING 10.0.0.2 from 10.0.0.1 h1-eth0
Unicast reply from 10.0.0.2 [42:75:EB:67:61:F6]  4.324ms
Unicast reply from 10.0.0.2 [42:75:EB:67:61:F6]  4.957ms
Unicast reply from 10.0.0.2 [42:75:EB:67:61:F6]  4.928ms
Unicast reply from 10.0.0.2 [42:75:EB:67:61:F6]  4.834ms
```

Hint: The priority of this flow rule MUST be higher than that of initially installed flow rule (40000), but not greater than 65535.

Part 2: Install Flow Rules (3/3)

- Install flow rules to forward IPv4 packets
  - Match Fields
    - IPv4 destination address
  - Actions
    - Output from port, forwarding IPv4 packets to hosts
- Verify the flow rules your installed

```
mininet> h1 ping h2                    # send ICMP request
```

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=9.00 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.54 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.188 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.075 ms
```

Hint:

1. *Switch may remove flow rules installed previously after a period of time.*
2. *Match fields may have dependency; please refer to OpenFlow spec v1.4.0.*

Part 3: Create Topology with Broadcast Storm

- Steps:
  1. Create a topology that may cause a **“Broadcast Storm”**.
  2. Install flow rules on switches of the network.
  3. Send packets from a host to another host.
  4. Observe link status of the network and the CPUs utilization of VM

- Do NOT activate any other APPs, except for those initially activated by ONOS
- Describe what you have observed and explain why the broadcast storm occurred.
- Hand in both Topology file (*.py) and flow rule files (*.json)

Hint: ONOS would initially install several flow rules.

Naming Convention

- Use the following convention to name the files created in both part 2 and part 3.
  1. Python script for the topology: topo_<studentID>.py
  2. JSON files for flow rules: flows_s<i>-<j>_<studentID>.json
     - "i" is the switch number
     - "j" is the flow rule number, starting from 1, on a switch.

     e.g.

File Name	Meaning
flows_s1-1_0748787.json	#1 flow rule to install on s1
flows_s1-2_0748787.json	#2 flow rule to install on s1
flows_s2-1_0748787.json	#1 flow rule to install on s2

Part 4: Trace ReactiveForwarding

- Activate only “org.onosproject.fwd” and other initially activated APPs.
- Use Mininet default topology and let h1 ping h2.
- Observe what happens in data and control planes
  - From the time when h1 pings h2 until h2 receives the first ICMP request
  - Write down each operation made by data and control planes
- Please refer to the ONOS ReactiveForwarding application
  - Source Code

Report

About Submission

Submission (1/2)

- Files:
  - A **report**: **project2_<studentID>.pdf**
    1. Part 1: Answers to those three questions
    2. Part 2, Part 3 & Part 4:
       Take screenshots of your procedure and also explain in detail
    3. What you've learned or solved
  - **JSON files** for installing flow rules in part 2 and part 3
    - Please follow naming convention
  - A **Python script** for creating topology in part 3

Submission (2/2)

- Directory structure:
  - Create root folder: **project2_<studentID>**
  - In root folder, create part2 and part3 folders and move files (i.e. *.json, *.py) into the corresponding folders

    e.g.

```
project2_0748787/
├── part2
│   ├── flows_s1-1_0748787.json
│   ├── flows_s1-2_0748787.json
│   └── flows_s1-3_0748787.json
├── part3
│   ├── flows_s1-1_0748787.json
│   ├── flows_s1-2_0748787.json
│   ├── flows_s2-1_0748787.json
│   ├── flows_s3-1_0748787.json
│   └── topo_0748787.py
└── project2_0748787.pdf
```

  - Zip root folder: **project2_<studentID>.zip**
- Wrong file name or format will result in 10 points deduction

Q & A

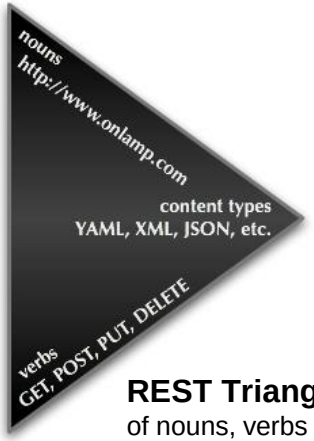
Thank you

References

- OpenFlow spec v1.4.0
  - https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf
- ONOS REST API
  - https://wiki.onosproject.org/display/ONOS/Appendix+B%3A+REST+API
- JSON Format for Installing Flow Rules
  - https://wiki.onosproject.org/display/ONOS/Flow+Rules

Appendix—REST (REpresentational State Transfer)

- REST is a **software architectural style** for creating Web services
- Architectural constraints:
  - Client-server architecture
  - Stateless
  - Cacheable
  - Uniform interface
  - Layered system



REST Triangle
of nouns, verbs and content types

Source: Soul & Shell Blog

- Allow us to access and manipulate web resources
  - Commonly we use HTTP method
    - Payload could be formatted in HTML, XML, JSON